October 2006

Self-paced Online Programming Instruction

Gayle L. Rambeau
Worcester Polytechnic Institute

Project Number: CS-KAL-0608

SELF-PACED ONLINE PROGRAMMING INSTRUCTION

A Major Qualifying Project

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

Gayle Rambeau

Date: 11 Oct 2006

Approved:

Professor Karen Lemone, Advisor

1. distance learning
2. education

Executive Summary

This project deals with the requirements of creating an effective distance learning system for basic programming instruction. The material component of this project centered on the defunct “Teach Yourself to Program” website constructed by my advisor, Prof. Karen Lemone, for the Frontier summer learning program at WPI. The key goal of the project was to revive this website, then analyze and improve it by applying useful metrics such as the concept of Transactional Distance.

To begin, the separate programming modules used in the original site were found, and combined into a new site titled “Teach Yourself Programming!” Then, these modules were analyzed and organized into a difficulty spectrum, along with minor interface changes to more closely support distance learning concepts. Finally, the visual design of the old site was updated to more closely match the conceptual design of the new site—difficulty tabs were added, as well as essential feedback tools such as a bulletin board system organized by programming language module.

The site was originally hosted on the webspace of the project team, and has now been moved to permanent residence on the project advisor’s webhost. This project was presented to the public during the WPI Fall Admissions Open House in a presentation for prospective freshmen.

Table of Contents

Abstract

The Teach Yourself Programming site was originally used to teach students basic programming skills in a WPI summer program. This project has reconstructed and redesigned it to fit essential distance learning principles in the hopes of encouraging its repurposing.

I. Introduction

As the saturation of technology continues to increase, prospective employees are finding that having basic programming skills is a highly-valued differentiating factor from other applicants when seeking new jobs or better positions. However, if an employee has not had the opportunity to take formal classes at a college or university, they may find it hard to gain the programming instruction they seek. To assist these non-traditional students, some institutions have turned to computerized instruction as a means of catering to the individual needs of students who may not be able to commit to normal structured classroom instruction. New "distance learning" programs are being studied and refined to allow busy professionals to better receive training on their own time, in the locations of their own choosing, home or office.

Every year, WPI hosts a summer program known as "Frontiers" for prospective students still in high school. Participants choose "majors" and spend two weeks exploring topics in their given subject area. For several years, students in the Computer Science program completed much of their coursework through various self-paced coding tutorials. These tutorials covered several languages, gradually increasing in their complexity, with the overall goal of providing a program for several different ability levels in the same classroom and with the same instructor.

The Frontiers program in Computer Science has since moved to other teaching formats. The code tutorials remain, but server moves and time have separated many of the files and parts of the original site are now defunct. As a

result, what were once valuable teaching tools have faded into obscurity and disrepair. However, this project seeks to repurpose old Frontiers materials to have a continued application outside of the program. The self-paced aspects, as well as the graduated difficulty of the modules themselves, lend themselves well to a non-traditional classroom. With further research, it might be possible to apply these program materials to distance learning and mixed-background classroom settings. Sites of this type currently exist on the internet, but none have the unique features of the former Frontiers modules.

The goal of my project is to address both the aforementioned question of adaptation and suitability, and the more practical problem of the current site's shortcomings, which are discussed in more detail following this introductory chapter. To accomplish this, I will first focus on the physical site, consolidating the scattered teaching modules and transferring them to a more permanent webspace. The site will also be redesigned to tie the modules together visually—the desired effect is that of an entire learning program rather than a collection of related resources from different sources. After the superficial work, I will standardize the modules somewhat by adding documentation and supplemental materials to those that did not include it. The overall goal of this process is to create a useful learning tool for computer science novices, both at WPI and elsewhere.

II. Background

In 2005, over 97 percent of public universities offered at least some coursework online as opposed to traditional classroom instruction (Wright). This rapidly increasing adoption of web-based teaching strategies (online coursework enrollments rose nearly 19% in only a year between 2003 and 2004) (Foster) is reflected in the growing body of professional articles and studies on the subject. In considering these sources, there are a few immediately attractive methods of approach valid to this project. Both supportive case studies and research, as well as dissenting views from educators and experts come together to paint a more complete picture of distance learning, as well as its relevance and application to the Teach Yourself Programming website.

The outlook on web-based classrooms was not always so bright. The London-based *Guardian* newspaper profiled the failure of one of the earliest major publicly-supported online instruction programs, known as "UkeU" (Hoare). In the early days of the new millennium, this UK government program was heralded as the new wave in education. But by 2003, the program was failing, plagued by high dropout rates. It was eventually abandoned and the torch passed to another research body, but the conditions of its demise are perhaps more interesting to study than the history of its run.

A few important lessons may be found in the fall of UkeU for anyone seeking to construct a distance learning system, even one on a much smaller scale such as the website component of this project. First is a caution of balance. UkeU and its contemporaries stood at the cutting edge of technology;

however, in this case that was not the unblemished advantage it might otherwise seem. While the presentation was top-of-the-line, students complained that it lacked a human component, effectively failing to be engaging. And the standards set by programs like UkeU were often too high to be replicated by other interested institutions, shutting out large sectors of the academic population who might otherwise have developed their own complimentary programs. This is not to downplay the importance of good technical design, but rather to highlight the need for an interface design and tone that retains a friendly, humanistic focus. As with all effective teaching styles, the interaction with students should be based on something more than just the novelty of a new toy.

Despite early setbacks, online instruction in many forms flourished and expanded in recent years. With this expansion comes an entirely new avenue for examination—the growing investment of private businesses in distance learning strategies not primarily intended for college students. Both academic and commercial concerns see non-traditional students as the untapped resource to further non-traditional learning, outside of a university environment. The *Journal of Commerce* examines this focus in more detail (Biederman). As with many innovations involving the Internet, non-classroom instruction on a large scale began as a side note to traditional college courses. The previously mentioned UkeU and its partner institutions sought mostly to connect university courses for an expanded student body, but one still enrolled full-time in an institution of higher learning.

While continuing education for professionals and those seeking to expand their skill sets was offered by many colleges and universities, the costs associated with it were in some cases too high for prospective students to bear. In addition to tuition, expenses such as extra transportation in commuting or lost work hours factored into a situation that discouraged many busy workers from taking advantage of existing classes. Distance learning is increasingly popular among these workers due to the features that characterize most online instructional programs. A distance learning program that seeks to effectively cater to non-traditional students would do well to emphasize features such as flexibility; a class that does not have meeting times and in which work is self-paced can be better tackled by a professional with a busy schedule. Online courses can also be better tailored towards the needs of individual students, either through self-pacing strategies like those used in Teach Yourself Programming, or through lesson plans developed in conjunction with a teacher.

Any movement of this magnitude generates a reasonable amount of scrutiny from reputable sources, and distance learning is no different. As previously noted, early distance learning systems did not enjoy universal success due in part to what students perceived as a detachment from human interaction. In January 2005, Salon.com [http://www.salon.com] published a critique of the current state of distance learning initiatives, and in addition to this complaint identified a few more common weaknesses (Wright). In the drive to commercially adapt distance learning techniques to reach wider audiences, in some cases the aforementioned detachment has been noted by educators as well as students.

Some educators feel that without a higher standard of interaction between students and professors, motivation to learn is negatively impacted and knowledge gained is reduced. Some further believe that it trivializes the learning process altogether.

It is criticisms in this vein that identify what may be a key constraint for this particular project. The Teach Yourself Programming website was originally developed to serve as an adjunct to classroom instruction provided for the Frontiers summer program. During its active run, it was used exactly as recommended by educational professionals, as a supplement. Direct professor and teaching assistant interaction was provided daily while using the site as a pacing guide. However, the Frontiers program no longer uses the website, and has no plans to readopt it at the conclusion of this project. Nor is the project currently associated with a specific class at WPI, to serve in the same supplemental manner. The standard operation of Teach Yourself Programming after the completion of this Major Qualifying Project will be as a self-guided learning tool, possibly without direct interaction with an instructor. Compensating for a loss in instructor support will be another feature to be kept in mind during the design phase of this project, even perhaps an avenue of further study for future projects.

Closely linked to the concept of instructional support is the theory of "transactional distance" (Moore). First developed by Michael Moore in 1973 and extended by other educational researchers since that time, it summarizes that the two key factors in successful independent learning are structure and

dialogue. Structure refers to the designed framework of the class itself, while dialogue covers the interaction between instructor and student. Transactional distance is a spectrum which indicates the "distance" between student and instructor in a given classroom or instructional program. As structure increases and dialogue falls, the transactional distance increases, while when the roles are reversed the distance falls. The effect of my design on transactional distance, especially in the areas of instructor dialogue, will be a key factor in my analysis of the finished system design.

Literature dealing with the current views of distance learning by professionals and academics, both in favor and in opposition, is a good starting point for gauging issues that may arise during the re-launch of Teach Yourself Programming. To expand, it seems appropriate to shift focus into more specific facets of distance learning. To this end, this project's research moved from the expression of public opinion to the evaluation of existing distance learning programs.

The EuroTraining program is a distance learning network sponsored by the European Commission, offering both its own courses and serving as a listing for other, smaller training centers. In a study conducted by Illyefalvi-Vitéz, et al. the EuroTraining network was examined and key successful features were noted. While EuroTraining differs from Teach Yourself Programming in scale and overall intent, these features may still be adapted for use in improving the experience of the latter.

The study begins by evaluating some weaknesses of the EuroTraining member programs. Some are superficial, and by extension could be considered "assumed" in terms of Teach Yourself Programming—broken links and related issues fall under this category. More interesting to this project are issues such as cluttered conceptual layouts, which made it difficult for the researchers to locate course materials. A second feature of particular importance to this project was a lack of clear user level/target audience guidance, essential to a system that provides multiple levels of instructional concepts.

Following the evaluation process, those involved in the study developed a proposed "ideal" distance learning course. Again, features highlighted in the study seem to be promising in terms of the design of this project. Navigation should be simple and obvious. Users should be as close to performance feedback as possible at all times, in the case of EuroTraining through self-tests (Teach Yourself Programming largely combines instruction with immediate feedback through interactive programs, marking a slightly different approach). A final suggestion revisits concerns raised by the study of earlier literature resources—a connection to human feedback, in this case a tutor or course instructor. This remains a key constraint within my own project, the lack of means and scope to provide for feedback of this type.

A second research study of a specific online learning system, in this case the CoMMIT system developed at the University of Pittsburgh (Lautenbacher), provides further insight into the ingredients of a successful web-based instruction program. This study utilized user testing on several different versions of the

program's interface, in an attempt to refine the existing system for optimal usability. Key findings were expressed in areas such as interface layout, in which students responded most positively to multi-framed interfaces that kept navigation visible at all times. User preferences tended towards being provided with more choices, preferring tiered menu structures where all options are available from the start. Another intriguing observation was a user preference for "metaphor"; students reacted most favorably to interface design that fit their mental models of other situations—in this case a blackboard, in the case of Teach Yourself Programming perhaps an update of the existing "desk" navigation system.

Teach Yourself Programming is not a large-scale distance learning network. It is comprised of only four learning modules, and for the purposes of this project is not associated with any classroom instructional course. The literature briefly introduced here deals with systems of a much larger scope, but fundamentally I believe that the concepts introduced in this body of contemporary opinion and research have no less application to a small-scale system as they do to a large one. They identify areas of focus, such as target audiences and students of multiple ability levels. They identify possible constraints of this project, such as the relative weakness of systems not associated with a live instructor providing feedback. Finally, the lessons learned from larger systems such as EuroTech and CoMMIT can usefully translate to projects of any size, even relatively modest distance learning programs such as Teach Yourself Programming. Interface design and learning style are essential considerations of

this project, and the existing literature discussed here provides a basic staging point for further development.

III. Design and Implementation

Before beginning a project of any size, it is a good idea to develop a strategy with which to proceed. In this instance, I have approached my eventual goal of a redesigned Teach Yourself Programming by accomplishing several subgoals, detailed below:

- Located the individual learning modules from the old site and retrieved them.
- Redesigned the site visually
- Evaluated the current stepwise ranking of the learning modules by ability
- Ensured that all modules are properly documented
- Took any other steps required to ensure that the site was expandable

As mentioned in the introduction to this project, the old Frontiers teaching material has fallen victim to several server changes. The first order of business was to locate the materials I wished to include in the new Teach Yourself Programming—which required the preliminary step of deciding which materials were essential to the project. After consultation with the "client" (in this situation, my project advisor, the original creator of the site), four learning modules were deemed to be useful. They needed to be consolidated in a single location to continue with the project.

Next, the site itself received a visual redesign. This redesign included more artistic considerations, such as color scheme and graphics, as well as the interface concepts introduced by existing studies on the subject of interface design and learning, both in general and in relation to web-based learning. The ultimate goal was to produce a site that both looked professional and evoked the spirit of the old Frontiers website, while following a framework developed from research into effective interface designs for learning.

In connection with the visual redesign, conceptual design was also reevaluated. The Teach Yourself Programming system was based around a stepwise ranking of modules by increasing difficulties. This ranking was explored, and reordered if necessary. Since the original intent of the site was to provide a simultaneous online teaching experience for multiple experience levels, this ranking was key to creating a relevant and functional site.

Also highly contributing to the usefulness of the site as a teaching tool are adequate teaching materials. Some of the modules provided their own integrated help systems, while others provide separate HTML documentation. I reviewed the outside documentation and supplemented or replaced it as needed.

Finally, I recognize that my project has some key constraints placed upon it due to time and scope. It is to this end that I considered myself a collaborator in the project—it is my hope that the work I began will continue to be improved upon by future projects. To conclude my design, I ensured that these constraints were well documented, and that the design of the site was open-ended enough as to not discourage further extension.

The Modules

The individual learning modules are the basic materials on which Teach Yourself Programming is founded. Each module introduces a different programming language, beginning with the most basic teaching language and working up to Java and HTML instruction. Since each module is such a large part of both the teaching system and this project, it will be useful to have a basic familiarity with each before venturing deeper into the details of the project. The following introductions to each module have been adapted from past Frontiers course materials. (Lemone, Frontiers 2001)

RoBOTL

RoBOTL is the beginning module in the Frontiers progression. It is a basic Object-Oriented (OO) language, based around the movements and actions of “bot” objects. Students learn basic code structure, such as numbered loops and “if” statements, as well as using pre-defined functions for actions. The teaching component of the module is an applet running the RoBOTL world to respond to commands, and documentation explaining how to perform basic bot tasks such as instantiation and program flow. As the module progresses, additional OO concepts such as inheritance are gradually introduced.

Tea

Tea is a basic object oriented language developed as an MQP by Yong Li. The module used in Teach Yourself Programming was developed as a follow-up MQP by students Leena Razzaq and Khue Huynh. It covers many of the same concepts as JavaBOTL, the next module on the website, but without the robotics metaphor. The Tea module contains a guided tutorial for each concept paired with a code environment applet that allows students to watch the code run. Following each tutorial a self-test is included. The Tea module is currently functioning on the academic website of one of its creators ("Tea: Learning to Program"), but the module was redesigned for use in Teach Yourself Programming.

JavaBOTL

JavaBOTL is an extension of RoBOTL, created through the union of multiple project groups, both MQP and IQP (a full listing is provided in the "Credits" section of Teach Yourself Programming). As the name would suggest, it connects the basic OO concepts of RoBOTL with the syntax of Java, in preparation for the step up to full Java. As with the other modules, students are guided through an HTML tutorial and given self-tests at the completion of certain sections.

Java

Unlike the RoBOTL and Tea modules, the Java module does not include its own applet-based interpreter. Students follow HTML instructions and view pre-made examples, while presumably using an external Java interpreter to construct their own code. It should be noted here that since contextual supplements have been lost in transfer, it may be useful to my goal of a self-contained system to write basic instructions on how to obtain a suitable, WPI-favored Java IDE such as Eclipse.

HTML Instruction

Like the Java module, the HTML module does not come with a simulated environment. Furthermore, it is designed mostly as a compliment to the Java module, as a way to create a staging area for the Java applets created during that program. It may be wise during the design period to evaluate combining these two modules into a more intuitive web publishing module.

Visual Design

To begin, we consider the existing Teach Yourself Programming website (Lemone, “Teach Yourself Programming”). The basic design consists of a two-frame system, with the left frame a small navigational bar, the right a content window. The right content window loads with an image map that also serves as

a second navigational panel. The picture used in the image map is of a desk (the site creator's) overlooking a forested area, which is where the inspiration for the site's color scheme, dark green and gray, is drawn.

Several design issues are immediately apparent. Superficially, the navigation bar contains a short introduction to the site that references three modules, while five are listed below it. More important to consider are the issues revolving around navigation within the site. This is first encountered on the front page, where a user is presented both with the navigational bar, containing links and descriptions of the available modules, and the aforementioned image map in the content frame. While visually appealing, the image map seems to be redundant—there is no description provided for its links aside from a visual representation, such as a robot or a cup of coffee. I find the concept of a "desk" as a metaphor for system navigation intriguing, but the presentation may be better served by relocating or repurposing this image or one similar.

The second navigational issue is encountered when clicking on some of the module links in the navigation bar. The user is taken to the front page for the module, which is expected, but the navigation bar also reloads into a different page. In addition, the bar loaded is not simply a subnavigation bar for the specific module, but a reduced module listing of the same navigational bar from the front page. This is perhaps a result of combining several modules with their own webspaces with an existing site. This is also suggested by the front pages of each working module, which load not with an introduction to the module but with a single image linked to the introduction. These intermediate layers

increase the time and actions needed to enter the module, without adding a significant amount of value or clarification.

To begin the visual design of Teach Yourself Programming, it might be indicated to consider the matter of intended audience for the moment. Different audiences demand different visual approach strategies—a younger audience would be engaged by bright colors and interesting shapes, while an older audience might feel patronized by these same features. The original Teach Yourself Programming was designed with Frontiers students in mind, so the intended audience was near to college age. While not for use in Frontiers, I feel it is safe to assume that the target audiences of this system will be either college students, directed to the site as part of their coursework, or self-motivated older learners, perhaps reaching the site by search engine. A flashy visual design for this audience may not be the best choice. Instead, a clean, professional looking approach to visual design would probably be the most successful.

Fig. 3.1 – Original navigation graphic

Fig. 3.2 – New desk graphic

The inspiration for the color scheme of the original site is drawn from the forest image used throughout. I have chosen to continue this color selection, focusing on greens and blues to correspond with my own “desk picture”, selected to echo the original site.

For the frame layout in the original site, a visual conflict occurs when loading a split-pane module such as Tea, where the screen becomes three consecutive vertical frames. A more visually appealing strategy would be to move the side navigation bar to the top of the screen, better encapsulating split-panel modules.

Next, the issues raised in the preceding site overview must be addressed. For the redundant front page design, the issue might be best resolved not by

removing the image map, but by removing information in the navigation bar. Since the navigation bar has been moved to a top aligned position, it is less cluttered to have this header contain only a site logo and a single line of text links to each module. The front page will be modified to include introductory site material, and the image map will now be accompanied by the descriptions of each module. In this manner, the clever metaphor concept is preserved, but given a non-trivial purpose in the site's design.

In the briefly discussed case study of the CoMMIT system at the University of Pittsburg, researchers discovered that students reacted most favorably to systems where navigation was transparent at all times (Lautenbacher). The menu choices used to reach a given tier should be visually retraceable. In the original site, once a module was entered the navigation bar changed, but not to sub-module information. In this system, it is essential for student clarity that a student not only be able to move from module to module clearly (through the top navigation bar), but through the parts of an individual module in a manner other than the inline links provided at the end of each tutorial page. To this end, when a student enters a module, the main navigation bar does not reload. However, a second subnavigation bar appears below the first, providing a text link to each "chapter" of the module. In this manner, students can see both levels of organization, module and chapter, at all times.

While some module chapters do include subchapters, it was concluded that adding a third navigational system to represent them might add unneeded complexity rather than increasing clarity.

Now that the weaknesses in the original site have been established in the area of site design, we turn our attention to the new Teach Yourself Programming site, somewhat introduced above. While the original design called for an ocean-toned site design, the final design incorporated only some of these colors. Instead, in keeping with the changes to conceptual design described below, the difficulty ranking of modules was extended into color scheme. The top nav-bar was replaced by a visual "tabbed" system, shown here.

Fig. 3.3 – The navigation bar

As can be clearly seen, the "Home" tab and link colors follow the ocean-themed color scheme, but the module tabs have the color of their respective difficulties. In this instance, it one of many trade-offs between visual design and conceptual design made during the design process.

Conceptual Design

Now that the basic framework for information presentation has been established, it becomes time to turn attention to the conceptual design of the information. As noted in the goals introduced at the beginning of the design chapter, one of the key elements of this step is evaluating the order of the

existing modules to determine whether they currently represent the best manner of increasing complexity.

The current order of the modules is RoBOTL, Tea, JavaBOTL, Java, and HTML. RoBOTL, as the very simplest module, is clearly a good choice for the learner with no previous programming experience. However, the next step in the progression is somewhat less clear. The conflict becomes apparent if one completes both the Tea and the JavaBOTL modules—much of the information presented is the same, just approached differently. This might speak towards the elimination of one of the modules, but each has features that set them apart. Tea distances itself from the "robot" metaphor, instead using simplified code that resembles Java structure to complete traditional programming tasks, mostly mathematical. JavaBOTL focuses more on "trivial" robot applications, but also has a structure that is closer to Java than Tea's (for example, JavaBOTL programs open with public static void main… though it is not explained in the module why this is so.) There is no clear module to eliminate, and perhaps not even a justification for elimination.

There is just cause, however, to support a reordering of modules. While it does not echo the exact syntax of Java as JavaBOTL does, Tea is closer in structure, and the problems presented by the Tea tutorials are more traditional programming problems like the ones found in the Java module. JavaBOTL focuses less on traditional programming than on the application of programming concepts to its own robot functions and framework, in addition to being very similar to RoBOTL for obvious reasons. A logical solution would be a new

module order of RoBOTL, JavaBOTL, Tea, and Java. This module ranking was the one adopted in the final site design.

A second conceptual issue arises when studying the Java and HTML modules. At first glance, they seem incomplete—the Java module requires a knowledge of HTML and basic publishing concepts (such as the use of WPI's Unix systems or Network Drive Mapping) to test applets, while the HTML module references applets frequently without providing any for students who may have skipped the Java module. Explanation can be found in the intended use of the current site; Frontiers students followed a two-part schedule of modules rather than a linear one (Lemone, Frontiers 2001). In the mornings, they learned programming concepts through the earlier modules, and in the afternoons they focused on learning the HTML modules. In this manner, they were learning multiple modules concurrently rather than linearly, which is a deviation from the intended progression suggested by Teach Yourself Programming.

A possible solution to this issue would be to simulate this concurrent learning on a smaller scale. Since the Java module requires HTML knowledge, and the HTML module references applets that may not have been created if a student did not complete the Java module, it follows that these two modules could be combined into one comprehensive module. This module, gathered under the title of "Web Programming", would begin by introducing beginning concepts needed for both modules, such as where to obtain a suitable Java IDE and links to the latest versions of Java. Although the audience for the site is not restricted only to WPI students, it is assumed that WPI students might be a

primary audience, therefore the Frontiers materials relating to navigating the WPI Unix system and links to network drive mapping on CCC computers will be included. Non-WPI students will have to do their own research to determine how they will handle making live websites with the module.

After the introductory material, the beginning part of the HTML module will be added. This will be enough to cover basic tasks such as setting up pages and adding text—enough to guarantee that the student will have a place to add their applet when they create it using the information from the Java module included next in sequence. After the Java module, the HTML chapters will conclude, and with them the final module of Teach Yourself Programming. At this stage, a list of links for further information about programming, HTML, Java applets, and perhaps even WPI might be helpful.

Once the new site was developed, it became clear that completely rewriting the HTML tutorials was somewhat out of the scope of this project. The documentation related to compiling Java applets was generalized somewhat for a non-WPI audience, but mostly only to the extent that students were advised to research Eclipse or other Java IDEs on the web. As mentioned later in the conclusion, a possible avenue for project expansion might be to create a new sub-module on this topic.

Supplementing Existing Documentation

The three initial modules (RoBOTL, JavaBOTL, Tea) are not particularly Frontiers-specific. They each come complete with their own HTML documentation, which would seem to need no further editing than being changed to fit the color scheme and layout of the new site. The Java and HTML modules, combined as described above, will need more extensive work. The references to Frontiers will be removed, while not harmful they are extraneous to the current usage of the system. The physical merging of the two modules will need to be accomplished, as well as a merging of their chapter systems to preserve clarity for students.

Additional documentation will be needed to create a more robust system. Currently, credit for each module is only provided on a by-module basis, if the creator of the module included it at all. In the interests of making sure each module contributor gets proper credit, a separate credits page will be added with this information. As mentioned in the visual design section, a front page will be added in place of the image used currently. The front page will contain the image map, a description of each module along with a difficulty estimation for each, a suggestion on how to proceed through the system, and links not pertaining to the modules, such as the credits page. A history page will also be linked to this page, listing a brief background on each language module prior to its inclusion in the Teach Yourself Programming website.

In the final site, most of the above is reproduced as described. It was decided that beyond the extensive credits section for each, the "homegrown" WPI teaching languages such as RoBOTL and Tea did not need history sections; thus only a brief history of Java is reproduced in this section. The Credits section has been updated as much as possible; it became clear during the project that many more project teams worked on the teaching languages than previously indicated. The team believes that all participants have been given correct credit for their contributions.

The Teach Yourself Programming website is reproduced in more detail in the "Results" section.

Extensibility

Though this issue will be more comprehensively discussed in both the analysis and conclusion chapters, it is important to keep in mind at all stages of the development of this system that it is an incomplete work. There are key areas which can and should be improved in the future. The affect that this fact has on the final design of the system is small in this iteration, mostly it influenced the decision not to include all non-module text links in the header bar. I also maintained a separate "designer archive" with design documentation, a copy of this project report, and editable design elements such as the .psd files for each graphical element in the site design. A small difficulty in my project arose from

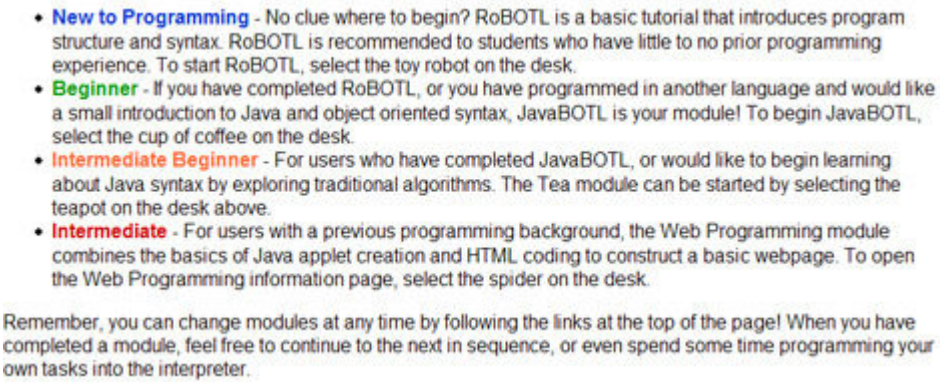
needing to adapt all the modules from their publicly available finished states, even in the case of images, and this archive should help smooth the process for future students. Should a student or instructor decide to extend the Teach Yourself Programming website, the designer archive will allow them to work from the foundation of my work, not just its visible output. The archive will be available upon request, either through contacting me or Professor Lemone, my project advisor.

IV. Results

Home

Fig. 4.1 – The Home tab of the navigation toolbar

As described in the Design phase, the user enters the site and first can observe the tabbed bar above. The bar is an imagemap, so clicking each of the tabs loads the page and sub-toolbars for each language module. The main page is loaded below the navigation bar in a separate frame. The conceptual ordering of difficulties continues in the module description on the main page.



- **New to Programming** - No clue where to begin? RoBOTL is a basic tutorial that introduces program structure and syntax. RoBOTL is recommended to students who have little to no prior programming experience. To start RoBOTL, select the toy robot on the desk.
- **Beginner** - If you have completed RoBOTL, or you have programmed in another language and would like a small introduction to Java and object oriented syntax, JavaBOTL is your module! To begin JavaBOTL, select the cup of coffee on the desk.
- **Intermediate Beginner** - For users who have completed JavaBOTL, or would like to begin learning about Java syntax by exploring traditional algorithms. The Tea module can be started by selecting the teapot on the desk above.
- **Intermediate** - For users with a previous programming background, the Web Programming module combines the basics of Java applet creation and HTML coding to construct a basic webpage. To open the Web Programming information page, select the spider on the desk.

Remember, you can change modules at any time by following the links at the top of the page! When you have completed a module, feel free to continue to the next in sequence, or even spend some time programming your own tasks into the interpreter.

Fig. 4.2 – The stepwise module description

The rest of the main page is dedicated to the supplemental documentation previously described. First is an invitation for visitors to take part in an on-going user study, first introduced at the MQP presentation described later. Below this, an important component of the project's transactional distance measures, the

interactive Bulletin Board System (BBS). The BBS, due to WPI code execution constraints, is hosted offsite using the free board hosting system InvisionFree [www.invisionfree.com]. Following the BBS link is a link to the Credits page, with all available language and module attributions, then the Information page which gives a brief summary of the history of Java for interested students. The contact info listed feeds to a centralized email account, to which both project team and advisor have access. As described in the "Extensibility" section, information for this account will be provided to all future project teams through the design archive.

Module Example: Tea

The Tea module was specifically noted in the beginning site analysis to cause visual discord. In the original site, the vertical navigation bar at the side of the screen resulted in a layout of three vertical tiles, as illustrated below.

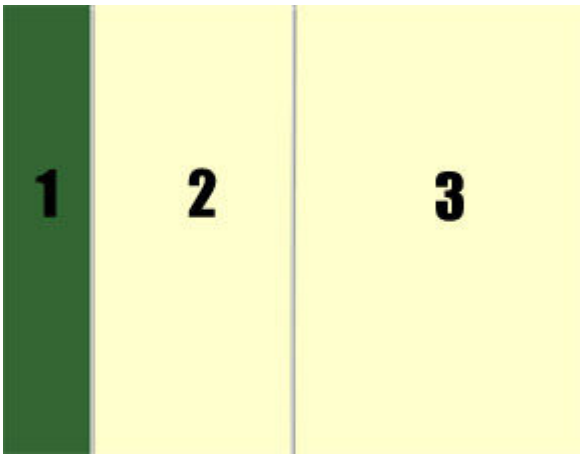


Fig. 4.3 – Tea Tiling

Fig. 4.4 – A more sensible Tea table

The construction of each module in the system echoes that of Tea, it is not needed to list them all here individually. For each module save RoBOTL, a special case, the main page consists of the colored navigation map, a smaller sub-bar of text links to each contained lesson, and a display panel in which the student moves through individual lessons. RoBOTL differs somewhat, in that the RoBOTL interpreter implementation chosen for the system exists within its own completely self-contained Java applet. In this case the module loads only a container page for the applet, which pops up its own window.

Now that the reader has received a basic overview of the results of the project's physical design, we move to more conceptual concerns. Our primary focus: given the opinions and concerns outlined in the literature review, has the new Teach Yourself Programming site met the challenges inherent in designing a distance learning system?

We learned from the study of the COMMIT system that students respond to several key design areas (Lautenbacher). Transparent navigation is chief among these—a student must at all times be able to move through the different structural layers of the system. If one refers back to Figure 4.4, it is clear that from a specific Tea lesson, the student can move to the index of all Tea lessons, a specific new lesson, and the rest of the programming language modules, all from the same page view. The rest of the programming modules are constructed in much the same manner. In our estimation, this is a reasonable fulfillment of the transparent navigation standard. Additional navigation assistance is provided through the use of a navigation metaphor borrowed from the old site, a desk with clickable images representing each programming module.

Perhaps the most important concept considered in the design was the metric of "transactional distance". If the reader will recall, transactional distance described the relationship between dialog and structure, and how they influence the interaction between student and teacher in the given distance learning system. This was of particular concern to our project because it identified a key constraint – there is no instructor associated with Teach Yourself Programming. From the metric, we are given to understand that a low-dialogue system like this

one must then be high in structure. Through navigation and restructuring, we believe we have accomplished this. The concern then becomes how to *simulate* instructor interaction where little exists. As a whole, the modules are not very feedback oriented—in Tea and RoBOTL errors are indicated, but not closely debugged. In other modules the only error feedback is received by checking student answers against a completed project key. It would seem that the most useful addition to the system would be a place where students could receive programming help on a per-module, per-question basis. To this end, we added a BBS attached to the site. The board is divided into forums by module, and students may create topics relating to their specific questions in each module. While the board has no dedicated monitor, there has been an informal agreement made by project team and advisor to make sporadic visits to the board. In this manner, we believe that the transactional distance has been shortened somewhat. Perhaps in the future, extensions to the system might be added to further reduce the transactional distance and create a more effective distance learning system.

V. Conclusion

One could say that this project began when I attended the Frontiers summer program 6 years ago as a rising senior. In addition to cementing my decision to apply to WPI, I also came into personal contact with most of the modules used in this system. When I was offered the opportunity to once again work with the system, I was eager to begin work on this project as the capstone of my WPI experience in computer science. Progress on the project was slowed somewhat by personal difficulties during my original project term, and work was extended into A Term of 2006. I feel that this was somewhat to the project's benefit—I was given the opportunity to present my system to prospective WPI students during the Fall Open House, perhaps paving the way for one of them to some day take up my work on this system.

As this statement suggests, work on Teach Yourself Programming is far from over. There are, as I feel, many opportunities for MQP and IQP projects relating to the website still existing. To this end, I created the design archive, to increase the extensibility of the overall project. Someone wishing to add a module to the group of existing modules, for instance, would merely have to add a new tab to the overhead navigation bar and update links, after determining where in the difficulty spectrum the module lies. Beyond this, other opportunities such as making the site completely compliant with HTML standards, updating the site to meet other metrics and performing user studies, adding components such as instructor chats or support to decrease transactional distance, or updating

specific modules such as the HTML module to train students in CSS are all valid and useful extensions of the Teach Yourself Programming website.

My only hope for the project now that my work into it has been completed is that it has the capability of introducing non-traditional students to the programming concepts that stand to become so important in the near future. I hope that students who extend my design share this same desire, and that the "old Frontiers site" finds its purpose anew.

Bibliography

Anderson, Linda. "Technology leads the way as online learning comes of age." *The Financial Times: Report – Business Education*. 20 Mar 2006. p. 1.

Biederman, David. "Going the distance." *Journal of Commerce.* 3 Oct 2005. p. 16.

Fisher, Mark. "Distance learning makes students grow fonder of college." *Dayton Daily News.* 28 June 2006. p. A10.

Giouvanakis, Thanasis; Samaras, Haido; Tarabanis, Konstantinos. "Designing a Pedagogically Sound Web-Based Interface: The Critical Role of Prior Knowledge." *ED-MEDIA 2001 World Conference on Educational Multimedia, Hypermedia & Telecommunications. Proceedings.* 2001-06. p. 592-597.

Hoare, Stephen. "Distance learning: Breaking with convention: Universities are at last reaping the benefits of research into online learning." *The Guardian: Education Pages.* 1 Nov 2005. p. 31.

Huynh, Khue; Razzaq, Leena. "Tea: Learning to Program." 2001. Accessed online at http://users.wpi.edu/~leenar/MQP/intro.html"

Illyefalvi-Vitez, Z.; Harsanyi, G.; Balogh, B.; Civera, P. "Analysis of distance learning courses offered by the EuroTraining course directory." *Electronic Components and Technology, ECTC '05. Proceedings:* Vol 2. 31 May - 3 June 2005. p. 1909 – 1915.

Lautenbacher, Glenn E.; Mahling, Dirk E. "Interface Design Issues for Web-Based Collaborative Learning Systems." *WebNet 97 World Conference of the WWW, Internet & Intranet Proceedings.* Nov 1997. p. 2-8.

Lemone, Karen. *Frontiers: 2001 Computer Science Program*. 2001.

Lemone, Karen. "Teach Yourself Programming." *Teach Yourself Programming*. Accessed online at http://web.cs.wpi.edu/~kal/robotl/teachyourself.html.

Moore, Michael G. "Towards a theory of independent learning and teaching." *Journal of Higher Education*: Volume 44, Issue 9. 1973. p. 661-679.

Pahwa, A.; Gruenbacher, D.M.; Starrett, S.K.; Morcos, M.M. "Distance learning for power professionals: virtual classrooms allow students flexibility in

location and time." *Power and Energy Magazine, IEEE:* Volume 3, Issue 1. Jan-Feb 2005. p. 53 – 58.

"PROFNET WIRE: Education & Science: Distance Learning." *PR Newswire US*. 15 Mar 2005.

Solomon, Gwen. "Shaping E-Learning Policy." *Technology & Learning*. May 2005. p. 26-29.

Wright. Alex. "From ivory tower to academic sweatshop." *Salon.com.* 26 Jan 2005. Accessed online at http://www.salon.com.